REMARKS

By this response, claims 1-22 remain pending in the same fashion as previously or originally presented. Substantively, all claims stand rejected as anticipated by Angelo 5,944,821 or as obvious in view of the combination of Angelo and Safadi 6,742,121.[1]

According to the Examiner, Angelo anticipates claims 1-4, 6-10, 12-17, 19-20 and 22 because, *inter alia*, Angelo "calculat[es] a score associated with the executable code [at a time] when the executable code is initially or shortly thereafter loaded into an operating system." *Underlining added, page 2, final ¶, 5-19-05 Final Rejection.* Relative to independent claim 21, the Examiner contends Angelo discloses "a first score associated with an executable code [at a time] when the executable code is initially loaded into an operating system' in (Col 11 lines 50-60, and Col 12 lines 1-10)." *Underlining added, page 6, first ¶, 5-19-05 Final Rejection.* In other words, the Examiner relies on Angelo for at least teaching scores being associated with executable code "at a time when the executable code is initially loaded into an operating system."

To the contrary, **Angelo never discloses** scores or calculations of scores associated with executable code at a time when the code is initially or shortly thereafter loaded into an

[1] Earlier, all claims stood rejected as anticipated by Tate 5,991,774. Despite the original claims all requiring a score and a subsequent (or second) score, and all amended claims reciting a more specific subset of the subsequent (or second) score, the Examiner has, nonetheless, made this rejection Final and changed it to depend wholly on Angelo and Safadi, not Tate. The Applicant reminds the Examiner that premature Final rejections are disfavored when the Applicant "switch[es] from one subject matter to another in the claims" or when the Examiner switches "from one set of references to another." *MPEP §706.07.* However, because the Applicant has not switched subject matter, just simply narrowed the focus of the previously presented claim limitations related to subsequent scores, the Applicant contends it is the Examiner's switch of cited references that is tending to defeat the attainment of disposition on this matter. The Applicant submits that if Tate was a sufficient reference during the first search, it should still be sufficient. Likewise, if Angelo and Safadi are now sufficient references, the Examiner should have uncovered them during the first search because the claims are now simply narrower subsets of the original, *see supra.* Further, just because the Examiner asserts the Applicant's amendments "necessitated the new grounds of rejection" (*Paragraph numbered 20, Page 6, 5-19-05 Final Rejection*), it does not mean the assertion is true. Rather, the Applicant contends the Examiner's assertion has no underlying facts exist to support it. ***To this end, the Applicant believes the Final rejection is premature and respectfully requests its withdrawal.***

operating system. Rather, Angelo calculates hash values of programs prior to or before the program being loaded into memory and executed. To this end, the Applicant respectfully requests reconsideration of all the pending claims.

According to Angelo, a secure hash table is created and "stored in protected [SMM] memory." *Col. 4, l. 34.* In the table, a secure hash value resides "for each program [or application] that the user wants to track." *Col. 4, l. 33.* Then, when users want to run the program or application,

> a system management interrupt (SMI) is generated. The SMI places the computer system in a system management mode, causing an SMI handler routine to be executed. The SMI handler first generates a current hash value for the program to be executed. Next, the SMI handler checks the stored hash table for an entry for the secured application. If a hash value entry is found, it is compared with the newly-calculated hash value for the secured application. In the event the two values match, the integrity of the application is guaranteed . . . If the two values do not match, the user is alerted to the discrepancy and may be given the option to update or override the stored hash table entry by entering an administrative password. *Col. 4, l. 56 - col 5., l. 5.*

It should be appreciated, however, that all instances of Angelo's hash value comparisons occur at a time prior to the executable code being installed in an operating system. More precisely, the above-quoted language recites a newly-calculated hash value for a secured application that is compared to a hash value entry of the table. Then, "[i]n the event the two values match, the integrity of the application is guaranteed and ***[thereafter]***

it is loaded into memory and executed."[2] *Emphasis added, col. 4, ll. 64-66.* Continuing, at *col. 4, l. 66 et seq.*, "[f]or security-sensitive applications, the entire application or a portion of it is loaded into system management mode memory (hereinafter "SMM" memory) ***prior to*** running the execution." *Emphasis added.*

Also, as seen below in Figure 3, Angelo generates the SMI (step 300), enters the SMM via the CPU (step 302), generates a hash value (step 304), validates the stored hash table (step 306), checks entries in the hash table and verifies same (steps 308-316) and then loads the program into memory and executes it (step 318).

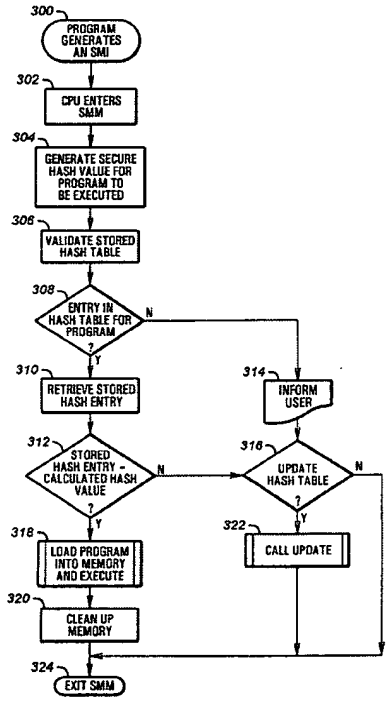

FIG. 3

From the usage of antecedent language in the steps of the above flow diagram, it should be appreciated that step 304 requires the generation of a secure hash value for "a

[2] Alternatively, at *col. 12, ll. 31-34*, Angelo recites as follows: "[i]n the event the two values match, the integrity of the application is guaranteed and it is [subsequently] loaded into either system management mode memory or normal memory and executed."

program to be executed." Then, at step 318, it is this same "program to be executed" that is subsequently "loaded into memory and executed." At *col. 9, ll. 6-8*, relative to Figure 2, Angelo reiterates this concept by stating the hash algorithm "securely register[s] and verif[ies] the integrity of software applications prior to execution." *Underlining added.* In other words, Angelo teaches hash value generation and comparison at a time prior to or before loading executable code (having its hash value generated) into an operating system.

In contrast, the pending claims relate (in one aspect) to scores or calculations of scores at a time when executable code "is initially or shortly thereafter"[3] loaded into an operating system. This, the Applicant respectfully submits, is antithetical to Angelo's "prior to" teaching and Angelo cannot then, as a matter of law, anticipate the claims. Angelo (alone or in combination with other references) cannot render them obvious either. Bear in mind, because Angelo teaches hash value comparisons "prior to" step 318 of LOAD PROGRAM INTO MEMORY AND EXECUTE, Angelo touts his technique as having "the additional advantage of being operating system independent." *Col. 5, ll. 18-19.*

Alternatively, Angelo teaches a further embodiment of his patent as "a modification to the loader" (*col. 11, ll. 51-52*) and the modified loader "verifies the program's hash value prior to execution." *Col. 11, ll. 55-56*). In turn, the loader is parenthetically defined as "(a part of the operating system that puts programs into memory for execution)." *Col. 11, ll. 53-55.* By rearranging the terms of the previous sentence, you can see the loader is 1) part of the operating system; and 2) that which puts programs into memory for execution. Since the

[3] This claim language is expressly found in claim 1. Of course, the other independent claims have related language but their slight variations change the overall scope of the claims. The Applicant also touts aspects of the instant invention as including this timing of loading code. Namely, "an aspect of the invention is to provide methods of detecting executable code which has been altered while in memory. By performing calculations against the executable image (e.g., code and data) during the initial loading of the executable code to an operating system to generate an initial score, and by performing subsequent calculations to generate subsequent scores associated with the executable image, a determination may be made as to whether the executable code has been altered." *Applicant's Specification, p. 4, ll. 4-10.*

loader is modified to verify the hash value of the program to be executed "prior to execution," the program is not then yet part of the memory. As a result, Angelo cannot be used to anticipate the pending claims which precisely require the antithetical score/score calculation of executable code at a time of initial loading (or shortly thereafter) into the operating system. Bear in mind, Angelo states his invention "builds on a trusted boot facility such as that described in the SAFESTART patent." *Col. 11, ll. 60-62.* In turn, SAFESTART is that which occurs in "User DOS, if installed." *Col. 4, l. 10.*

Still alternatively, Angelo disparages prior art solutions that attempt to solve problems of code modification detection via software products and/or in other than protected memory areas, such as SMM memory. For example,

> [i]n some earlier systems, a secure hash value is calculated and stored for newly installed software. Thereafter, when the computer is turned on again, the stored hash value is compared to a newly calculated value. If a discrepancy is found, the user is alerted. A main disadvantage with this method is that the integrity assessment codes must be stored on the hard disk, thus making the codes themselves susceptible to attack by malicious code. Reverse-engineering a modification detection code, while difficult, is not a mathematically intractable problem. Thus, software-only protective products can offer only limited insurance against the attack of malicious code, due mainly to architectural weakness present in most computer systems. A potential solution is to embed the modification detection code in a permanent read-only memory device, but this can make system reconfiguration quite difficult. *Col. 2, l. 60 - col. 3, l. 8.*

Thus, Angelo teaches away from the instant invention, especially in instances, such as claims 21 and 22, requiring computer readable medium.

Consequently, Angelo and the instant invention have antithetical teachings and reconsideration of all pending claims is respectfully requested.

Relative to claims 5, 11, 18 and 21, the Examiner contends Safadi supplies the missing teaching of "receiving one or more additional scores periodically on the executable code and disabling the executable code if the subsequent score is not equal" (*Page 5, final ¶, 5-19-05 Final Rejection*); or "discloses the 'Detection of Suspect software objects and signatures after failed authentication'." (*Page 6, first ¶, 5-19-05 Final Rejection*). Support in Safadi is given exclusively as "Col 8, lines 5-22." *Page 5, final ¶ and page 6, first ¶, 5-19-05 Final Rejection.*

In its entirety, Safadi simply recites at *col. 8, ll. 5-22*:

> FIG. 3 is a flowchart illustrating the steps performed at a set-top terminal upon invoking a software object. A download request occurs at step 100. Thereafter, in step 110, the BIOS, operating system and/or the Java Virtual Machine (JVM), when requiring the download or the use of a software object, call(s) the set-top CA routine for an authentication and authorization check. The use or launch of the object is allowed only if the check passes. The CA check is facilitated by the secure processor provided in the set-top. In addition, a lifetime feature may be implemented, wherein the secure processor records the object lifetime and checks it for expiration, starting for example with first use (i.e., the first time the secure processor was engaged in authenticating and authorizing the object). If the object is expired, the secure processor may interrupt the operating system or JVM to disable/delete the object(s). If any of the checks fail, the set-top terminal may log the results to report back to the access controller. Again, this feature is preferably implemented using a combination of software and hardware functions.

However, the Applicant does not dispute that Safadi teaches an iterative process of checking authorizations. Also, the Applicant does not dispute that the prior art is replete with instances of usage where iterations of a variety of software checks occur. However, to suggest that because Safadi broadly teaches iterations of authorization checks it somehow (in combination with Angelo) renders claims 5, 11, 18 and 21 obvious, is to over-generalize or over-simplify the relationship of the claim terms.

As seen on pages 5 and 6 of this response, claim 21 (for example) precisely requires executable code, embodied in a computer readable medium, to have 1) an unaltered original format; 2) a first score associated therewith at a time when the executable code is initially loaded into an operating system; 3) a second score associated therewith at a period of time subsequent to when the executable code was initially loaded and operable to be compared with the first score to determine if the executable code has been altered since the initial load; and 4) an altered unoriginal format if the first and second scores do not equal.

In other words, claim 21 requires a precise interaction of code format and scores relative to loading times of the code in an operating system. The Applicant submits that rejecting such precise claims as obvious in view of Safadi's broad teaching, that iterative authorization checks for software occur, is akin to arguing that a broad genus anticipates or renders obvious a definitively more narrow species.

In contrast, the law has long provided, a "prior art reference that discloses a genus still does not inherently disclose all species within that broad category." *Metabolite Laboratories, Inc. V. Laboratory Corp. of America Holdings*, 71 USPQ2d 1081, 1091 (Fed. Cir. 2004)(The court also quoted from *Corning Glass Works v. Sumitomo Elec. USA, Inc.*, 868 F.2d 1251, 1262 [9 USPQ2d 1962] (Fed. Cir. 1989)("**Under [defendant's] theory, a claim to a genus would inherently disclose all species. We find [this] argument wholly meritless** [sic]. . . .")(Bold Added)).

Nonetheless, if Safadi arguably stands for the proposition asserted by the Examiner, nowhere does Angelo supply the missing teaching of "a first score associated with an executable code when the executable code is initially loaded into an operation system." *Page 6, first ¶, 5-19-05 Final Rejection.* By incorporation of the above-presented arguments, the Applicant submits Angelo antithetically teaches hash value comparisons at a time "prior to," not after, the loading and execution of programs. To this end, the Applicant requests reconsideration.

Alternatively, if Angelo arguably stands for the proposition asserted by the Examiner, nowhere does Safadi supply the missing teaching necessary to render the claims obvious. Namely, the above-quoted *col. 8, lines 5-22* of Safadi teach nothing to render obvious independent claim 21 or dependent claims 5, 11 or 18. They simply teach an iterative process of authorization checks for software. Nowhere do they hint at scores associated with code, formats of code or when scores of code are calculated or obtained, let alone when scores are obtained relative to an operating system. In fact, Safadi is completely silent as to the salient features of all the pending claims.

To the Examiner's assertion that Safadi teaches "Detection of Suspect software objects and signatures after failed authentication' which includes a method above in (Col 8, lines 5-22)," the Applicant responds by questioning the relevance of the statement as to the arrangement of limitations in claim 21, for example, that recite:

> 21. Functional data used to validate executable code embodied in a computer readable medium, the data comprising:
>
> an unaltered original format;
>
> a first score associated with an executable code when the executable code is initially loaded into an operating system;

a second score associated with the executable code at a period of time subsequent to when the executable code was initially loaded and operable to be compared with the first score to determine if the executable code has been altered since the initial load; and

an altered unoriginal format if the first and second scores do not equal.

Respectfully, the Applicant reminds the Examiner that it is impermissible to utilize hindsight reconstruction when examining the claims. The proper test of obviousness is whether the differences between the invention and the prior art are such that "the subject matter as a whole would have been obvious at the time the invention was made" to a person skilled in the art. *Stratoflex Inc. V. Aeroquip Corp.*, 713 F.2d 1530, 1538 (Fed. Cir. 1983)(Underlining added). Bear in mind, the Applicant originally filed for patent protection on May 22, 2001. It is now more than four full years after filing. The Applicant also reminds the Examiner of caution expressed by the Court of Appeals for the Federal Circuit that "[d]etermination of obviousness can not be based on the hindsight combination of components selectively culled from the prior art to fit the parameters of the [] invention." *ATD Corp. v. Lydall, Inc.*, 159 f.3d 534, 536 (Fed. Cir. 1998).

Lastly, the entirety of the dependent claims are submitted as being patentable because of their direct of indirect dependence on one of claims 1, 8, 16, 21 or 22 having limitations discussed above. Additional reasons of patentability can be given but are being held in abeyance in anticipation of a Notice of Allowance.

Consequently, the Applicant submits that all claims are in a condition for allowance and requests a timely Notice of Allowance to be issued for same. ***To the extent any fees are***

due, although none are believed due, the undersigned authorizes the deduction from Deposit Account No. 11-0978.

Respectfully submitted,

KING & SCHICKLI, PLLC

Michael T. Sanderson
Registration No. 43,082

247 North Broadway
Lexington, Kentucky 40507
Phone: (859) 252-0889
Fax: (859) 252-0779

Certificate of Mailing
I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: MAIL STOP AFTER FINAL (AF), Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
on June 14 2005
Date 6-14-05 by Carolina Perdomo